

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-243035

(43)Date of publication of application : 02.09.1994

(51)Int.Cl.

G06F 12/08

G06F 13/00

G06F 15/16

(21)Application number : 05-324668

(71)Applicant : BULL HN INF SYST INC

(22)Date of filing : 22.12.1993

(72)Inventor : HUNTER JOHN C  
WERTZ JOHN A

(30)Priority

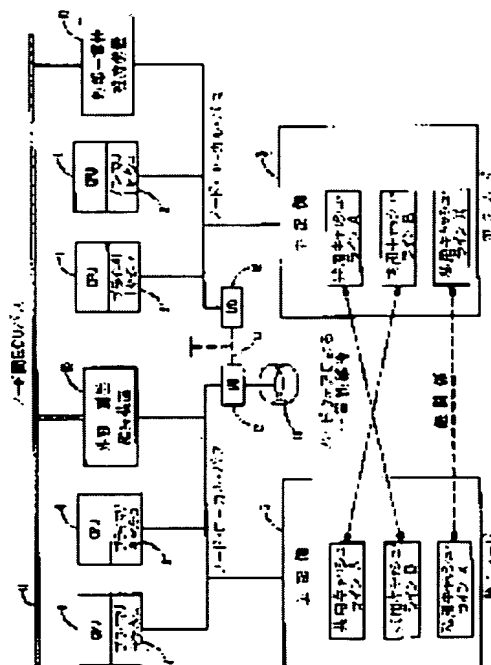
Priority number : 92 993884 Priority date : 23.12.1992 Priority country : US

## (54) GENERALIZED SHARED STORAGE IN CLUSTER ARCHITECTURE FOR COMPUTER SYSTEM

(57)Abstract:

PURPOSE: To improve the performance and generalization of the cluster architecture of a computer.

CONSTITUTION: A 'generalized shared storage' is introduced to a cluster architecture, and this generalized shared storage is held in a state with coherency by a coherency maintaining mechanism constituted of hardware which adds an operation to a shared object when the positioning or reference or the like of the shared object is operated. Each of plural nodes is provided with a central processing unit 1 having a primary cache 2, local main storage 3, outside coherency maintaining device (ECU) 10, and node local bus 4 connecting them. Those nodes are connected through an inter-node ECU bus 11. The ECU monitors the bus, detects a cache line request, and executes a coherency maintaining operation according to the state of the requested cache line. The rate of a private storage/shared storage can be arbitrarily and dynamically changed in each node in the cluster.



## LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than

the examiner's decision of rejection or  
application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of  
rejection]

[Date of requesting appeal against examiner's  
decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平6-243035

(43) 公開日 平成6年 (1994) 9月2日

(51) Int. Cl. <sup>B</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G06F 12/08		H 7608-5B		
13/00	355	7368-5B		
15/16	310	M 7429-5L		

審査請求 未請求 請求項の数9 (全 14 頁)

(21) 出願番号 特願平5-324668

(22) 出願日 平成5年 (1993) 12月22日

(31) 優先権主張番号 993884

(32) 優先日 1992年12月23日

(33) 優先権主張国 米国 (U S)

(71) 出願人 391031270  
ブル・エイチエヌ・インフォメーション・システムズ・インコーポレーテッド  
BULL HN INFORMATION SYSTEMS, INCORPORATED  
アメリカ合衆国マサチューセッツ州 01821, ビレリカ, テクノロジー・パーク (番地なし)

(72) 発明者 ジョン・シー・ハンター  
アメリカ合衆国アリゾナ州85021, フェニックス, ウェスト・エコー・レーン 121

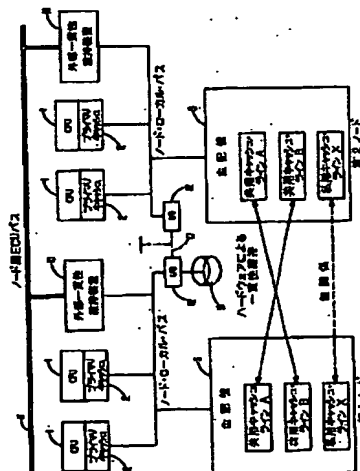
(74) 代理人 弁理士 湯浅 恭三 (外5名)  
最終頁に続く

(54) 【発明の名称】 コンピュータ・システムのためのクラスタ・アーキテクチャにおける一般化共用記憶

## (57) 【要約】

【目的】 コンピュータのクラスタ・アーキテクチャの性能及び汎用性を向上させる。

【構成】 クラスタ・アーキテクチャに「一般化共用記憶」を導入し、この一般化共用記憶は、共用オブジェクトの位置付けや参照等が行なわれたときにその共用オブジェクトに対して操作を加えるハードウェアで構成した一貫性維持機構によって、一貫性を有する状態に維持される。複数のノードの各々が、プライマリ・キャッシュ2を有する中央処理装置1と、ローカル主記憶3と、外部一貫性維持装置 (ECU) 10と、それらを結合するノード・ローカル・バス4とを備えている。ノード同士はノード間ECUバス11によって接続される。ECUはバスを監視してキャッシュ・ライン要求を検出し、要求されたキャッシュラインの状態に応じて一貫性維持動作を実行する。私用記憶/共用記憶の割合をクラスタ内の個別のノードにおいて任意に且つ動的に変化させることができる。





ドへ移転し、

C) 要求されたキャッシュ・ラインの状態が「共用」状態であったならば、「無効化」コマンドをその他の全てのノードへ送出することによって、その他の全てのノード内において、要求されたキャッシュ・ラインの状態を「無効」状態にセットさせる、ようにしてあることを特徴とする請求項2記載のコンピュータのクラスタ・アーキテクチャ。

【請求項4】 前記ノード間通信手段が、ノード間外部一貫性維持装置バスを含んでいることを特徴とする請求項1記載のコンピュータのクラスタ・アーキテクチャ。

【請求項5】 前記ノード間通信手段が、ノード間外部一貫性維持装置バスを含んでいることを特徴とする請求項2記載のコンピュータのクラスタ・アーキテクチャ。

【請求項6】 前記ノード間通信手段が、ノード間外部一貫性維持装置バスを含んでいることを特徴とする請求項3記載のコンピュータのクラスタ・アーキテクチャ。

【請求項7】 前記ノード間通信手段が、直接結合手段を含んでいることを特徴とする請求項1記載のコンピュータのクラスタ・アーキテクチャ。

【請求項8】 前記ノード間通信手段が、直接結合手段を含んでいることを特徴とする請求項2記載のコンピュータのクラスタ・アーキテクチャ。

【請求項9】 前記ノード間通信手段が、直接結合手段を含んでいることを特徴とする請求項3記載のコンピュータのクラスタ・アーキテクチャ。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明はデータ処理技術に関し、より詳しくは、クラスタ・アーキテクチャに用いる一般化共用記憶に関する。

【0002】

【従来の技術】 一般的に、クラスタ・アーキテクチャと呼ばれているコンピュータ・アーキテクチャでは、複数のノードの各々に、少なくとも1つの（そして通例は複数の）中央処理装置（CPU）と、ローカル記憶と、入出力装置（I/O）と、その他の要素とを装備している。良好な評価を受けているクラスタ・アーキテクチャのうちの1つに、物理記憶アドレス空間を恒久的に2つの領域に分割しておくようにしたクラスタ・アーキテクチャがあり、そこでは、分割した下半分の領域（最上位ビット＝「0」の領域）を、当該ノードに装備されているプロセッサだけがアクセスすることのできるローカル記憶とし、一方、分割した上半分の領域（最上位ビット＝「1」の領域）は共用記憶を画成するようにしており、この共用記憶は物理的に集中化してあって、全てのノードがそれにアクセスすることができる。ある共用変数の複数のコピーが別々のノードに保持されているときに、それら複数のコピーの間の一貫性を維持するためのハードウェア機構は装備されていない。そのため一貫性

の維持は完全にソフトウェアに任されている。

【0003】 以上のアーキテクチャを拡張したものが、「Bull NH 共用バッファ・アーキテクチャ」（以後、共用バッファ・アーキテクチャを「SBA」と略称する）であり、その拡張とは、物理的に集中化してある共用記憶に本来所属するデータの、ただしクラスタ内の別々のノードのキャッシュ記憶の中に保持されている、複数のコピーの間の一貫性（コヒーレンシー）を維持するためのハードウェア機構を導入したことにある。これによって、別々のノードが、共用データに対するある種の操作を互いに並行的に、より短いアクセス時間で実行できるようになっており、それによってアーキテクチャの性能が改善されている。

【0004】 SBAの変形構成の1つである「Bull NH 分散共用バッファ・アーキテクチャ」（以下、分散共用バッファ・アーキテクチャを「DSBA」と略称する）は、集中化した共用記憶に代えて分散した共用記憶を用いており、この構成形態は、性能及び可用性、それに既存のある種のコンピュータ製品との間の互換性という点に関して、多くの利点を有している。私用記憶と共用記憶との間のアドレス区分は固定されていて不変であるが、共用記憶を各コンピュータ・ノード内に複製してあるため、全てのノードが同時に読取りアクセスを行なうことができる。またハードウェアで構成した一貫性維持機構によって、分散した夫々の記憶の中のデータどうしの間の一貫性を維持するようにしている。

【0005】

【発明が解決しようとする課題】 公知のアーキテクチャの変形構成である上述のアーキテクチャは、それらのいずれもが、私用記憶と共用記憶との間の区分を固定して不変にしていた。この特性のために、クラスタの個々のノードの内部でも、また複数のノードの間でも、私用記憶及び共用記憶を構成するに際して、そのときどきアプリケーション・ミックス（アプリケーションの組合せ）の要求に適合するように、それら2通りの記憶を異なった大きさに構成することが（たとえ全く不可能でないにしても）困難であった。共用アプリケーションを実行するのに私用記憶を使用することはできず、一方、私用アプリケーションを実行するのに共用記憶を使用したならば、その私用アプリケーションは不必要なコヒーレンシー・トラフィックに煩わされる上に、完全性に関する問題を生じるおそれもある。本発明はこれらの制約を克服するものである。

【0006】 従って本発明の広い目的は、改良したクラスタ・アーキテクチャを提供することにある。

【0007】 本発明のより具体的な目的の1つは、任意の大きさとすることができクラスタ内の任意の個数のノードによって共用させることのできる不連続な共用オブジェクトを保持する一般化共用記憶を用いるようにしたクラスタ・アーキテクチャを提供することにある。

【0008】本発明の更に別の具体的な目的は、クラスタ内の全ての要素が共用記憶を極めて迅速に入手できるようなクラスタ・アーキテクチャを提供することにある。

【0009】

【課題を解決するための手段】要約して述べるならば、本発明の以上の目的並びにその他の目的は、クラスタ・アーキテクチャに「一般化共用記憶」を導入することによって達成される。この一般化共用記憶は、共用オブジェクトの位置付けや参照等が行なわれたときにその共用オブジェクトに対して操作を加えるハードウェアで構成した一貫性維持機構によって、一貫性を有する状態に維持される。これによって、私用記憶/共用記憶の割合をクラスタ内の個別のコンピュータ・ノードにおいて任意に且つ動的に変化させることができることから、アーキテクチャの性能と汎用性との両方が向上する。

【0010】

【実施例】先ず最初に、従来例に係るクラスタ・アーキテクチャを示した、図1のハイレベルのブロック図について説明する。図1に例示したシステムは、16個の中央処理装置(CPU)1を、各ノードが4個ずつのCPUを備えた4つのノードに編成した構成を有する。各CPUは、その他のシステム構成要素との間の通信に関して一貫性を維持する特性を有しており、また通常そうであるように、プライマリ・キャッシュ2を装備している。更に、各ノードはローカル共用記憶3を含んでおり、ローカル共用記憶3は、そのノード内の夫々のCPU1との間でプライマリ・キャッシュを介して通信を行なう際に、ノード・ローカルな通信システムを介してその通信を行ない、図示例におけるノード・ローカルな通信システムは、ノード・ローカル・バス4である。加えて、各々のノード・ローカル・バス4は、クラスタ共用記憶5にも結合している。ローカル共用記憶3の各々とクラスタ共用記憶5とは同一の大きさにしても良く、例えばそれらを同じ「2ギガバイト」とすることができる。記憶をアドレスするためのアドレス構造は、次のようなものであり、即ち、あるCPUが発した記憶アドレスの最上位ビット(MSB)が「0」であったならば、その記憶アドレスは、そのCPUが所属しているノードのローカル共用記憶の中に格納されている情報をアドレスするためのものであり、また、その記憶アドレスのMSBが「1」であったならば、その記憶アドレスは、クラスタ共用記憶の中に格納されている情報をアドレスするものであると定めてある。ノード間通信は1/O装置12及びノード間バス17を介して行なわれ、通信をするノードどうしの間の距離がかなりのものであるならば、通信バス20を用いれば良く、以上のことは全て、当業界においては公知の事柄である。

【0011】以上に説明したアーキテクチャは、適当な操作を行なうことによって、クラスタ共用記憶5を介し

てクラスタ内の全域において情報交換を行なえるという利点を有する。しかしながらこのアーキテクチャには、その利点と同時に、幾つかの短所も付随している。それら短所は例えば次のようなものである。

A) 私用記憶と共用記憶との区分が恒久的に固定されており、そのため、実行すべきアプリケーション・ミックスが、私用記憶を用いた方が良好に機能するアプリケーションと、共用記憶を用いた方が良好に機能するアプリケーションとの両方を含んでいる場合には、非常な非効率を生じることがある。

B) クラスタ共用記憶には、シングル・ポイント・オブ・フェイリヤが存在している。

C) 各々のノードごとに、そのノードのためのオペレーティング・システムを装備しておく必要がある。

D) クラスタ共用記憶への個々のアクセスの各々ごとに、全てのノードの記憶管理機構の同意を得る必要がある。

【0012】既述の如く、図1に示したアーキテクチャに変更を加えてシステム性能の漸増的改善を達成した変形構成態様が幾つか存在している。それらのうち、共用バッファ・アーキテクチャ(SBA)に該当する構成態様としては、集中化した共用記憶に本来所属するデータの、ただしクラスタ内の別々のノードの夫々のキャッシュ記憶の中に保持されている、複数のコピーどうしの間の一貫性を維持するハードウェア機構を装備したものである。これによって、別々のノードが、共用データに対する複数の操作を互いに並行的に、より短いアクセス時間で実行できるようになっているため、アーキテクチャの性能が向上している。また、分散共用バッファ・アーキテクチャ(DSBA)としては、集中化した共用記憶に代えて分散共用記憶を用いたものがあり、この構成形態は、性能及び可用度、それに既存のコンピュータ製品との間の互換性という点に関して、多くの利点を有している。私用記憶と共用記憶との間のアドレス区分は固定されていて不変であるが、共用記憶を各コンピュータ・ノード内に複製してあるため、全てのノードが同時に読取りアクセスを行なうことができる。またハードウェアで構成した一貫性維持機構によって、分散した夫々の記憶の中のデータどうしの間の一貫性を維持するようにしている。

【0013】これらに対して、本発明は、クラスタ・アーキテクチャに、根本的な新機軸を提供するものである。本発明を理解するには、その前に先ず、本発明の重要な一局面である「共用オブジェクト」の概念を明確に認識しておく必要がある。共用オブジェクトは、任意の大きさとすることができ、また、クラスタ内の任意の個数のノードの間で共用させることができる。複数の共用オブジェクトの集合体によって「一般化共用記憶」が構成され、この一般化共用記憶は、共用オブジェクトの位置付けや参照等が行なわれたときにその共用オブジェク

トに対して操作を加えるハードウェアで構成した一貫性維持機構によって、一貫性を有する状態に維持される。

【0014】共用オブジェクトは、共用し得るという性質を備えたオブジェクトであると定義することができる。共用オブジェクトは、クラスタ内の全てのノードに知られている。共用オブジェクトのエクス Tent (存続範囲) と、アクセスを共用することを許可されているプロセスとは、記述子によって明示される。共用オブジェクトの実現形態は、目的オペレーティング・システムの種類によって様々に異なり、ファイルの形態とされることもあれば (例えば「Bull NH 社」の「GCOS 8」オペレーティング・システム等の場合)、ストリームの形態とされることもある (例えば「UNIX」オペレーティング・システム等の場合)。共用オブジェクトは各ノードごとに個別に、そのノードの仮想空間内でインスタンス化され、それによって、異なったノードの夫々のプロセスによって共用され得るようになる。仮想空間内でのインスタンス化が行なわれたならば、共用オブジェクトは (例えばコマンドによって)、分散共用記憶を用いたクラスタの別々のノードの夫々の記憶の中で、各ノードごとに個別に、物理的にインスタンス化することができるようになる。集中共用クラスタ記憶の中では、ある1つの共用オブジェクトの幾つもの物理的なインスタンス化が行なわれることもあり、例えば、複数の「UNIX」プロセスの間で共用されるページをサポートする共用オブジェクト等がそうである。共用オブジェクトは以下のものを備えている。

- A) 任意長表示の一義的名標
- B) 固定長表示の一義的識別子 (UID)
- C) アドレス可能な内部要素
- D) 参照許可

【0015】DSBA環境に即して考察することが、共用オブジェクトに対するリアルタイムの操作を理解するための最も容易な方法である。図2を参照して説明すると、クラスタ内の各ノード (図を見易くするために、図2にはノードのうちの2つだけ示してあり、また各ノード内のCPU1のうちの2個だけを示してある) は、外部一貫性維持装置 (External Coherency Unit: ECU) 10を装備している。ECU10は (a) そのノードのノード・ローカル・バス4を監視しており、そのノード・ローカル・バス4上にその他のノードの中にも存在している「共用」キャッシュ・ラインに関するコマンドが送出されたときにそれを検出し、そして (b) それに該当するコマンドを検出したならば、条件に従ってそのコマンドを (そのキャッシュ・ラインを表わす一義的識別子を使用して) その他のノードへ例えば独立したノード間ECUバス11等を介して転送する。 (更にはポイント・ツウ・ポイント接続も可能であり、そのためには、各々のECUの中のディレクトリを用いてどのノードがコピーを保持しているかを追跡するようにする)。—

方、その他の全てのECUが (a) ノード間ECUバス11を監視しており、そのノード間ECUバス11上に、それら夫々のECUのノードのローカル記憶の中に存在しているキャッシュ・ラインに影響を及ぼすコマンドが送出されたときにそれを検出し、また (b) それに該当するコマンドを検出したならば、そのコマンドを、そのノードにおける物理タグへ変換し、そして (c) 変換したコマンドを、そのノードにおけるノード・ローカル・バス上へ送出する。概念説明のための図示の具体例では、ECU10を一般的な一貫性維持装置を用いて構成することも可能であり、その装置において、一義的識別子を有するキャッシュ・タグ・ディレクトリと、そのノードで現在インスタンス化されている各「共用」キャッシュ・ラインの状態を表わす一貫性状態 (例えば、変更済、専有、共用、無効) とを用いるようにすれば良い。

【0016】具体的な動作例について説明すると、例えばクラスタ内の、ある1つのノードの中のある1つのCPUが、ある1本の共用キャッシュ・ラインの内容を変更しようとする場合、そのCPUは、何よりも先ず最初に、専有所有権を獲得する必要がある。そのためのコマンドがローカルECU10によって監視されており、もし、そのノードが専有所有権を持っていなかったならば、そのコマンドはその他の全てのノードのECUへ転送される。すると、それらノードのうち、変更されていない有効なコピーを保持していた各ノードは、 (例えばそのノードのプライマリ・キャッシュ2、及び/または、ローカル・メイン記憶3の中にそれまで保持していた) 全てのローカル・コピーを無効化する。もし、あるノードがそのキャッシュ・ラインを「変更済」状態で保持していたならば、そのノードは、その「変更済」状態のキャッシュ・ラインを要求元ノードへ移転した上、その移転元ノードの中でそのキャッシュ・ラインを無効化する。これによって、その (移転された) キャッシュ・ラインが専有的に所有されて、更新可能な状態になる。以上の一連の一貫性維持作業により、共用オブジェクトを参照しているプロセスは、それがどのノードにおいて実行されているどのプロセスであっても、最新のデータを受け取ることができるように保証されている。

【0017】以上のアーキテクチャによれば、I/O装置12は、ある共用オブジェクトの全て或いは一部を保持することを割当てられているローカル物理記憶との間で適正に作業を行なうことができる。即ち、I/O装置12が共用記憶からの読込みを行なうときには、 (そのI/O装置12が装備されているノード以外の) その他のノードの共用記憶の中にあった最新データが自動的に移転される。また、I/O装置12が共用記憶への書込みを行なうときには、そのI/O装置12が装備されているノードにおける専有所有権が、データの到着と共に、自動的に獲得され、そして、I/O装置12がその

作業を完了した後は、いずれかのノードのプロセッサがそのデータを参照したならば、そのデータが自動的にそのノードの記憶へ移転される。この一般化共用記憶を採用することによって得られる利点は次のとおりである。・ノードの内部でもノードどうしの間でも物理記憶をより効率的に使用し得ることに関する利点：

(1) 物理的インスタンス化した私用記憶領域及び共用記憶領域が非連続領域であっても良く、それら領域の合計が物理記憶の大きさを超えない限り、それら領域を任意の大きさにすることができる。

(2) 同一時刻における夫々のノードの私用／共用の割合を互いに異ならせることができる。

(3) 記憶の合計の大きさを、単一の実オペレーティング・システムないしプラットフォームの物理的アドレス能力以上にすることができる。

・クラスタのソフトウェアが簡明化されることに関する利点：

(1) 共用記憶を各ノードごとに個別に管理するため、大域的記憶管理機構が不要であり、また、共用領域を管理するために複数のノードが互いに協働して作業する必要もない。

・可用性が向上することに関する利点：

(1) SBAの集中共用記憶に存在しているシングル・ポイント・オブ・フェイリヤが、ここには存在していない。

(2) あるノードが故障した場合に、そのノードの私用記憶を強制的に共用可能にすることにより、別のノードがその私用記憶領域にアクセスして故障からの回復を図ることができる。

・コヒーレンシー・トラフィックが軽減されることに関する利点：

(1) ノード間コヒーレンシー・トラフィックが発生するのは、非専有的に所有されている共用オブジェクトへの書き込みが行なわれるときと、存在していない共用オブジェクトが参照されるときとの、いずれかの場合に限られる。

(2) 共用オブジェクトのコピーを保持していないノードからは、コヒーレンシー・トラフィックが排除されている。

【0018】なお付言すると、この「分散した」一般化共用記憶の性能は「集中化した」共用記憶の性能を凌駕し得るものであり、なぜならここに提案した機構は、共用データがキャッシュ方式の高速性という利点を享受し得るようにすると同時に、伝統的なアドレス可能な記憶をそのまま残しているからである。即ち、この機構によれば、個別の複数のノードが、同じ共用変数の、ただしそれら夫々のノードがみずからの中に保持しているところの、複数のコピーを、各々のローカル・バスが発揮し得る限りの高速でもって互いに同時に読み取ることができるのである。これに対して、「Abrojo」やSB

Aに用いられている集中共用記憶は、同時に多数の読取り要求が寄せられた場合には、みずからがボトルネックとなってしまう。

【0019】「共用」し得るものか否かを検出するための基本的な方式として2通りの方式があり、その1つは「ハードウェア方式」、もう1つは「ソフトウェア方式」である。

ハードウェア方式：ページ・テーブル記述語 (Page Table Descriptor Word: PTW) に、あるページが共用ペ

10 ジであるか否かを表示するための1つのビット（「共用」ビット）を含ませておく。共用領域の参照が行なわれるときにはそのPTWがCPUの中に存在しており、このCPUがノード・ローカル・バス上へ「共用」ビットを送出することによって、そのノードに装備されているECUに、共用領域の参照が行なわれることを通知する。こうすることによって、ECUを、共用ページの物理タグに関するキャッシュとして動作させることができ、このECUキャッシュの中には、高頻度でアクセスされるタグだけを実際に保持しておくようにする。参照

20 頻度の低いタグはECUが記憶から取り出すようにしておく。

この方式の長所：簡明、高速、低コスト、そしてソフトウェア独立（即ち、ソフトウェアに左右されることがない）である。

この方式の短所：CPU及びバスのハードウェアに改造を施す必要がある。

【0020】ソフトウェア方式：現在インスタンス化されている共用ページのPTW（実際にはECUのエントリ）の全てをECU内のバッファ記憶の中に保持してお

30 き、このECUが、監視を行なうことによって（或いは、ECUどうしを直接接続してディレクトリを参照することによって）共用キャッシュ・ラインを検出するようにしておく。共用ページのインスタンス化または削除が行なわれる度に、記憶管理ソフトウェアが、ECU内のPTWを保持しているバッファに対して更新を施すようにする。

この方式の長所：中心的なシステム・ハードウェアに改造を施す必要がない。PTWに余分なビットを付加する必要がない。そして「既存のハードウェアにも装備し得

40 る可能性を有する」。

この方式の短所：記憶管理ソフトウェアに改造を施す必要があり、ハードウェアのコストが増大する。

【0021】これより、図3の、より詳細な具体例のブロック図について説明して行く。ECU10は連想ディレクトリ13を含んでおり、この連想ディレクトリ13の機能はノード・ローカル・バス4上のアドレス並びにノード間ECUバス11上のアドレスを常時監視することにある。ここでは、このディレクトリ13の中に、同じECUの中のローカル記憶3内に現在物理的にイン

50 スタンス化されている共用ページのの各々に1つずつが対



応した、ECUエントリが保持されているものとする。ここで図6を参照して簡単に説明しておくと、ECUエントリは、それが対応しているページのローカル物理アドレスを包含していると共に、そのページの中に含まれている複数本のキャッシュ・ライン（例えば、4096バイトの大きさのページの中には連続した64バイトから成るキャッシュ・ラインが64本存在している）の各々に対応した一義的識別子及び一貫性状態ビット（2～3ビット）のビット集合を包含している。1つのECUエントリの全体の大きさは24バイトから32バイトまでの間の大きさであり、これが1つの共用ページに対応する。

【0022】説明を再び図3に戻し、ECU10は、非共用のキャッシュ・ラインに関する要求のコマンドは無視する。そのためその種のコマンドは、そのまま、ローカル記憶3の非共用物理空間14の中の指定された記憶をアドレスすることになる。一方、ECU10が識別したコマンドが、共用キャッシュ・ラインに関するコマンドであった場合には、ECU10は、ECUディレクトリ16の中に格納されているその共用キャッシュ・ラインの一貫性状態（「変更済」、「専有」、「共用」、または「無効」）を調べることによって、一貫性を維持するためには何らかのノード間動作を行なうことが必要か否かを判定する。そして、ノード間動作が必要であると判定されたならば、物理アドレスを然るべき一義的識別子に変換した上で、その変換した識別子を、該当するコマンドと共に、ECUバスを介してクラスタ内の他のノードへ転送する。

【0023】一例として、あるCPUが、ある共用キャッシュ・ラインの専有コピーを獲得しようとする場合、そのCPUは、然るべきコマンド（例えばRTWコマンド等である。RTWコマンドとは「書き込みを前提とした読取り」コマンドの意味であり、これについては具体例のシステムに関連して後に更に詳細に説明する）を、そのCPUが所属しているノードのノード・ローカル・バス上に送出する。すると、そのノードのECU10が以下の動作のうちのいずれかを実行する。

(1) そのキャッシュ・ラインの状態が「専有」状態または「変更済」状態であったならば、そのキャッシュ・ラインをローカル記憶3から直接読み出して、要求元プロセスへ転送する。この場合、そのキャッシュ・ラインのコピーは他のどこにも存在していないため、遠隔動作は何も実行しない。

(2) そのキャッシュ・ラインの状態が「無効」状態であったならば、ノード間ECUバス11を介して他のノードへRTWコマンドを送出する。ある遠隔ノードが、そのキャッシュ・ラインを「専有」状態または「変更済」状態で保持していたならば、その遠隔ノードは、ノード間ECUバスを介してそのキャッシュ・ラインを要求元ノードへ転送してくる（即ち、移転が行なわれ

る）。また複数のノードが、そのキャッシュ・ラインを「共用」状態で保持していたならば、それら複数のノードの全てがそのキャッシュ・ラインを送出しようとする。しかしながら、ECUバスの標準的な優先権ロジックが、それら送出のうちの1つを選択してその他をキャンセルする。更に、全ての遠隔コピー（遠隔ノードに保持されているコピー）が「無効」状態にセットされる。

(3) そのキャッシュ・ラインの状態が「共用」状態であったならば、INVコマンドを他のノードへ送出することによって、他のノードが夫々みずからの中に保持しているそのキャッシュ・ラインのコピーを「無効」状態にセットするようにする（この場合には、要求元ノードの中に既に現在コピーが存在しているため、移転が行なわれる必要はない）。

【0024】バスからの入力とキャッシュ・ラインの状態との組合せが以上のものとは異なる様々なものである場合に、夫々に一貫性を維持するために必要となるその他のノード間動作も存在しているが、それらについては、後に要約して述べることにする。一貫性維持のための以上の処理手順は、ハードウェア・レベルで動作するものであり、そして、共用ページを参照するプロセスが、それがどのノードにおいて実行中のどのプロセスであっても、その共用ページの全体に互って最新のデータを見られるようにすることを保証するものである。

【0025】ページ・イン方式：経済性という観点からの要請に従うならば、例えばページ・フォールトを発生させるような参照（即ち、無効PTW）がなされたとき等に行なう、共用ページの物理的インスタンス化は、実際にその共用ページを参照するノードの中だけで行なわれることが望ましい。これに関して従来のシステムでは、記憶管理機構ないし記憶管理ソフトウェアが、物理ページ及びPTWのインスタンス化を行なった後に、I/O装置に、例えば従来のI/O通信チャネル17等を介してそのページの内容をページ・インさせるようにしていた。ところが、分散記憶システムの場合には、そのページの内容が既に他のノードの中に存在している可能性もある。そこで、記憶管理機構ないし記憶管理ソフトウェアは、通常のページのインスタンス化の作業を行なうに加えて、そのページの内容がどこかに存在していないかを調べるサーチも行なわせることが必要となる。これらのことを行なうには、(1) ECUの中に、全てのキャッシュ・ラインに「無効」のマーキングを施したECUページ・エントリを格納した後に、(2) ページ・フォールトを発生させた記憶位置への再度の参照を（今度は有効PTWによって）試みるようにする。これを行なったならば、その結果は(1) 要求されたキャッシュ・ラインが別のノードからその記憶の中のページ・フレームへ移転されるか、或いは、(2) そのページはどのノードの共用記憶の中にも存在していないと判定されるかの、いずれかとなる。結果が(2)となった場

合には、記憶管理ソフトウェアは、そのページを保持している記憶装置（例えばディスク装置）へのI/Oアクセス権をどのノードが持っているのかを判定した上で、そのノードがI/O動作を実行してそのディスク装置からそのノードみずからの記憶の中へそのページをページ・インすることを（例えば特別の割込等によって）要求する必要がある。そのI/O動作が完了したならば、本来実行していたソフトウェアを、問題となっているその参照が行なわれた点から再始動させることができるようになり、また夫々のノードのECUが、移転によってその参照を完了させることができるようになる。

【0026】ページ・アウト方式：夫々のノードにおけるページ置換の判断は、個々のノードごとに、他のノードからは独立して行なえるようにしておくことが望ましい。複数のノードにおいてインスタンス化されている共用ページの場合には、例えばそのページが極めて低頻度でしか参照されないノードがあるなら、そのノードのローカル記憶からは、そのページを除去できるようにしておくことが望まれる。そこで、ある長さの時間に亘って一度も参照されていない共用ページがあって、その共用ページが変更済のものであるならば、そのページのホーム・ノードへ（例えば特別の割込等によって）、そのページをディスク装置18（図3）へ書込むべきことを伝えることによって、そのページを除去できるようにしている。このとき、そのホーム・ノードの記憶管理ソフトウェアは、もし、そのページを物理的にインスタンス化したものがホーム・ノードの中に存在していなかったならば、先ずそのページの物理的インスタンス化を行ない、その後1/O動作を開始して、そのページの中の各々のキャッシュ・ラインを参照して行きながらそのページの最新の完成コピーを他のノードから集めるようにする。遠隔ノードの中のキャッシュ・ラインの一貫性状態は、以前のままに放置しておくようにしても良く、或いは、夫々の遠隔ノード内で「無効」状態にセットするようにしておいても良い。後者の方式とすれば、そのページが低頻度でしか参照されないノードにおいて、そのページを削除することが容易になる。（PTWが変更済ページであることを示している場合であっても、ECUに対して照会を発することによってそのページ内の全てのキャッシュ・ラインが無効であることを判定できるようにしてあれば、記憶管理ソフトウェアはそのページを安全に廃棄することができる）。これ以後、そのページを参照しようとするノードが現われたならば、そのノードは、ホーム・ノードの中に存在しているコピーの移転によって、そのページのデータを受け取ることになる。

【0027】これより、図4及び図5について説明する。これらの図は、CPU1と、そのCPU1のプライマリ・キャッシュ2と、ノード・ローカル・バス4と、ECU10と、ノード間ECUバス11との間のインターフェースを示した状態図であり、従って、同機種から

成る「GCOS 8」環境に組み込めるようにしたECU10を規定している状態図である。この「GCOS 8」環境で使われる様々な用語の意味は次のとおりである：

・第1レベルのバスとは、CPU1と、そのCPU1のプライマリ・キャッシュ2とを結合しているバスのことである。

・第2レベルのバスとは、ノード・ローカル・バス4のことである。

・第3レベルのバスとは、ノード間ECUバス11のことである。

・データ移動コマンドに関しては次のとおり：

I1—第1レベルのバスから受取ったデータ。

O1—第1レベルのバスへ出力されたデータ。

I2—第2レベルのバスから受取ったデータ。

O2—第2レベルのバスへ出力されたデータ。

I3—第3レベルのバスから受取ったデータ。

O3—第3レベルのバスへ出力されたデータ。

・インターフェース・コマンドに関しては次のとおり：

<RD1>—プロセッサへ読込む。

20 <WR1>—プロセッサから書込む。

<RAR1>—再書込み後読取りの原子動作。

・更に第1レベルのバス（図5）に関しては次のとおり：

<RD2>—第2レベルのバスからキャッシュ・ラインを要求する。

<RTW2>—専有所有権を確保した上でキャッシュ・ラインを読取る。

<INV2>—第1レベルの全てのプロセッサに対し、キャッシュ・ラインの無効化を命じる。

30 <WR2>—第2レベルのバス上へキャッシュ・ラインのデータを書出す。

・更に第2レベルのバス（図6）に関しては次のとおり：

<RD2>—第2レベルのバスからキャッシュ・ラインを要求する。

<RTW2>—専有所有権を確保した上で他のCPUからキャッシュ・ラインを読込む。

<INV2>—ローカル・バスである第2レベルのバス上の全てのキャッシュの中のキャッシュ・ラインを無効

40 化する。

<WR2>—プライマリ・キャッシュからキャッシュ・ラインを除去し、或いは、移転する。

<RD3>—第3レベルのバスからキャッシュ・ラインを要求する。

<RTW3>—専有所有権を確保した上で他のノードからキャッシュ・ラインを読込む。

<INV3>—全てのECUにキャッシュ・ラインの無効化を命じる。

50 <WR3>—第3レベルのバス上へキャッシュ・ラインを書出す。

【0028】当業者であれば容易に理解することであるが、本発明は階層化接続の階層数が何層であっても容易に適用し得るものであり、即ち、第4レベル、第5レベル、等々が存在している場合でも適用可能である。

【図面の簡単な説明】

【図1】従来例に係るクラスタ・アーキテクチャを示したハイレベルのブロック図である。

【図2】本発明に係るクラスタ・アーキテクチャを示したハイレベルのブロック図である。

【図3】本発明に係るクラスタ・アーキテクチャの外部一貫性維持装置（ECU）の構成要素の中間レベルのブロック図を特に示したより詳細なブロック図である。

【図4】一例として取り上げた具体的なコンピュータ・ファミリーにおけるプライマリ・キャッシュの状態遷移

図である。

【図5】図4と同一の具体的なコンピュータ・ファミリーに採用し得るようにした外部一貫性維持装置（ECU）の状態遷移図である。

【図6】外部一貫性維持装置の連想記憶のエントリの具体例を示した図である。

【符号の説明】

- 1 中央処理装置（CPU）
- 2 プライマリ・キャッシュ
- 3 ローカル共用記憶
- 4 ノード・ローカル・バス
- 10 外部一貫性維持装置（ECU）
- 11 ノード間ECUバス

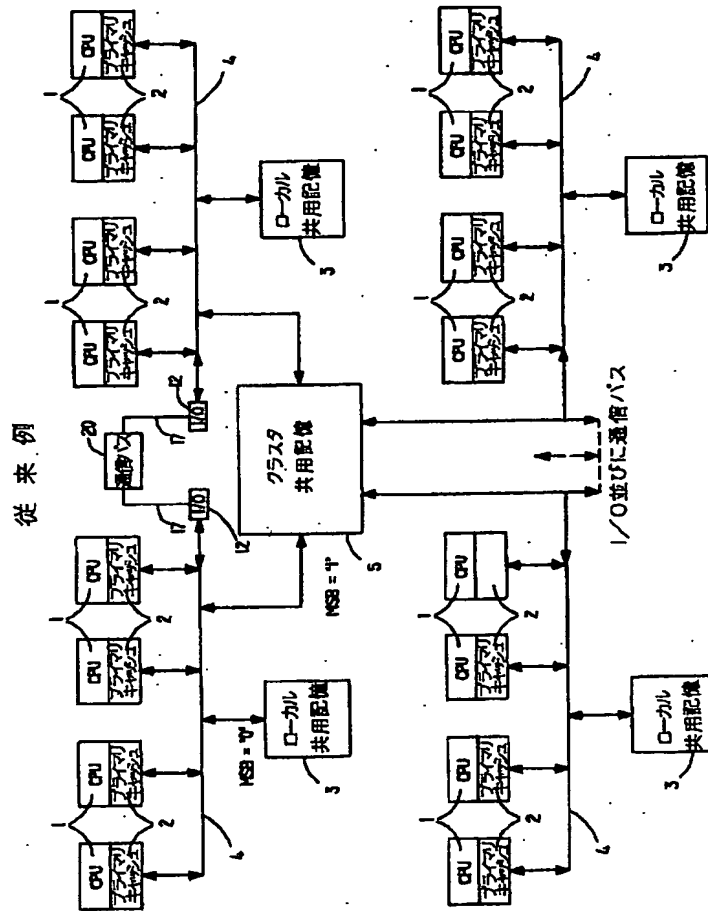
【図6】

外部一貫性維持装置エントリ（ECUエントリ）  
（各共用ページごとに1つずつ）

ノード内の物理ページ・タグ 一義的識別子	ページ内の各キャッシュ・ライン の一貫性状態
-------------------------	---------------------------

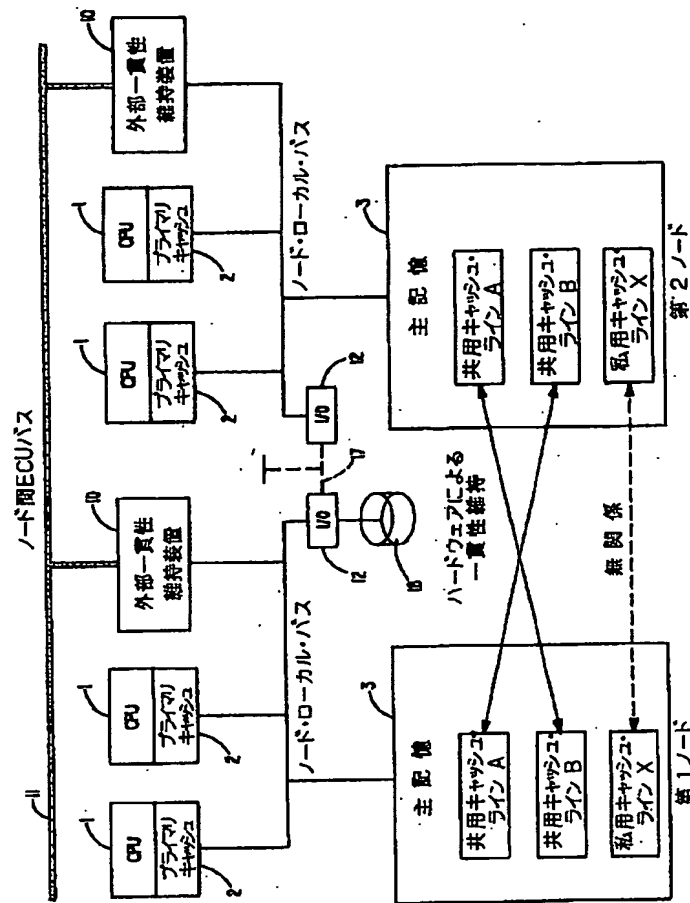
(10)

【図1】



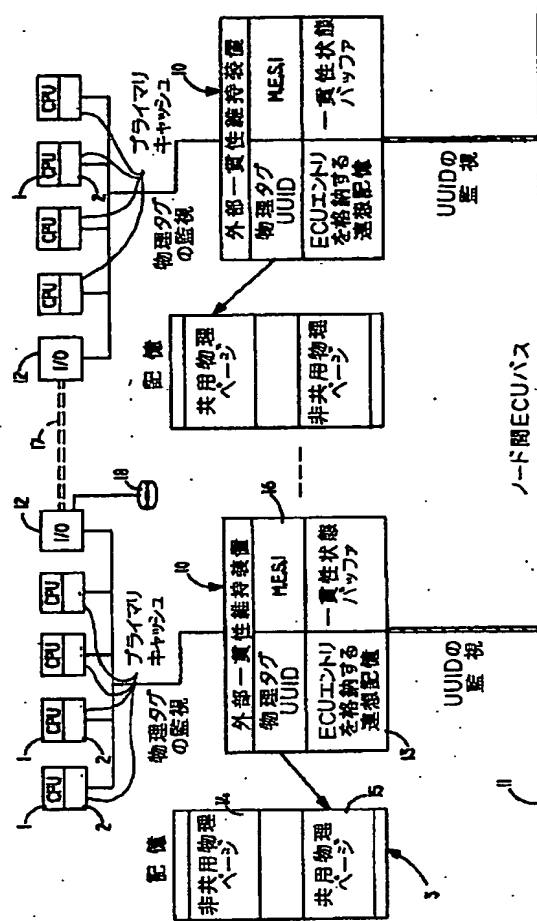
(11)

【図2】



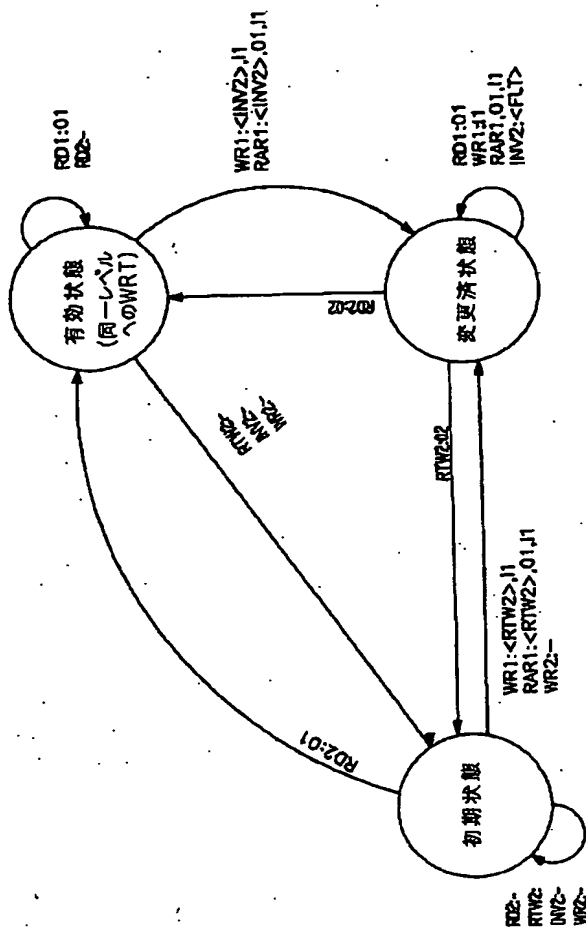
(12)

(図3)

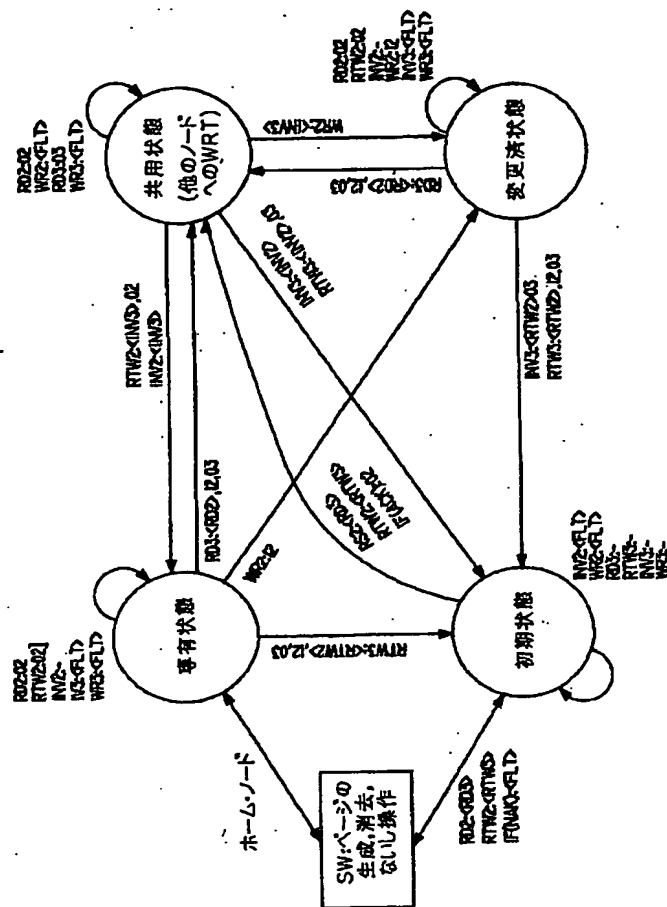


(13)

【図4】



【図5】



フロントページの続き

(72)発明者 ジョン・エイ・ワーツ  
 アメリカ合衆国アリゾナ州85023, フェニ  
 ックス, ノース・トゥエンティース・ドラ  
 イブ 17402